**3.Interogari multiple**

În capitolele anterioare am aflat cum putem afişa informaţii din baza de date, însă la fiecare rulare a unei comenzi **SELECT** am afişat date dintr-o singură tabelă.

Din fericire SQL oferă facilităţi pentru combinarea datelor din mai multe tabele şi afişarea lor într-un singur raport. O astfel de operaţie se numeşte **join**, sau **interogare multiplă**.

Pe parcursul acestui capitol vom folosi ca exemple tabela **Persoane** a cărei cheie primară este atributul **IdPersoana**,tabela **Firme** a cărei cheie primară este atributul **IdFirm**, şi tabela **Joburi** cu cheia primară **IdJob**. Presupunem că aceste tabele conţin următoarele înregistrări:

**Tabelul II.3.1.** Tabela **Persoane**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **IDPERSOANA** | **NUME** | **PRENUME** | **LOCALITATE** | **IDFIRM** | **IDJOB** |
| 1 | Ionescu | Gheorghe | Brasov | 22 | 5 |
| 2 | Vasilescu | Vasile | Cluj-Napoca | 15 | 1 |
| 3 | Popescu | Ioan | Bucuresti | 10 | 2 |
| 4 | Georgescu | Maria | Iasi | 30 | 6 |
| 5 | Marinescu | Angela | Sibiu | - | 3 |
| 6 | Antonescu | Elena | Sibiu | 10 | 1 |
| 7 | Bischin | Paraschin | Brasov | 15 | - |
| 8 | Olaru | Angela | Ploiesti | 22 | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabelul II.3.2.** Tabela **Firme**   |  |  |  | | --- | --- | --- | | **IdFirm** | **Nume** | **Localitate** | | 10 | SC Crisib SA | Sibiu | | 15 | SC SoftCom | Alba Iulia | | 20 | SC TimTip | Timisoara | | 22 | Brasoveanca | Brasov | | **Tabelul II.3.3.** Tabela **Joburi**   |  |  | | --- | --- | | **IdJob** | **Nume** | | 1 | Reprezentant Vanzari | | 2 | Manager | | 6 | Operator IT | | 3 | Programator | | 4 | Administrator | | 5 | Administrator retea | |

În Oracle există două moduri diferite de a scrie joinurile:

         Prima **metodă foloseşte sintaxa specifică Oracle**. În acest caz condiţiile de join sunt incluse în clauza **WHERE**. Această metodă este mai uşor de înţeles, însă are dezavantajul că în aceeaşi clauză **WHERE** se includ atât condiţiile de filtrare a înregistrărilor afişate cât şi condiţiile de join.

         A doua variantă **foloseşte sintaxa ANSI/ISO**, care este puţin mai greoaie, însă comenzile scrise folosind această sintaxă sunt portabile şi în alte SGBD-uri care folosesc limbajul SQL.

Indiferent de sintaxa folosită există mai multe moduri de legare a tabelelor şi anume:

         **Produsul cartezian** – leagă fiecare înregistrare dintr-o tabelă cu toate înregistrările din cealaltă tabelă.

         **Equijoin** – sunt legate două tabele cu ajutorul unei condiţii de egalitate

         **NonEquijoin** - în acest caz condiţia de join foloseşte alt operator decât operatorul de egalitatea

         **SelfJoin** – este legată o tabelă cu ea însăşi, e folosită de obicei în conjuncţie cu relaţiile recursive.

         **OuterJoin** – sunt o extensie a equijoinului, când pentru unele înregistrări dintr-o tabelă nu există corespondent în cealaltă tabelă, şi dorim ca aceste înregistrări fără corespondent să fie totuşi afişate.

**II.3.1. Produsul cartezian**

**a) Sintaxa Oracle**

După cum am precizat, acest tip de legătură între două tabele, va lega fiecare rând din prima tabelă cu fiecare rând din cea de a doua tabelă. De exemplu comanda:

**SELECT p.nume, p.prenume, f.nume**

**FROM persoane p, firme f**

Va afişa următoarele informaţii

**Tabelul II.3.4.** Produsul cartezian între tabelele **Persoane** şi **Firme**

| **Nume** | **Prenume** | **Nume** |
| --- | --- | --- |
| Ionescu | Gheorghe | SC Crisib SA |
| Vasilescu | Vasile | SC Crisib SA |
| Popescu | Ioan | SC Crisib SA |
| Georgescu | Maria | SC Crisib SA |
| Marinescu | Angela | SC Crisib SA |
| Antonescu | Elena | SC Crisib SA |
| Bischin | Paraschin | SC Crisib SA |
| Olaru | Angela | SC Crisib SA |
| Ionescu | Gheorghe | SC SoftCom |
| Vasilescu | Vasile | SC SoftCom |
| Popescu | Ioan | SC SoftCom |
| Georgescu | Maria | SC SoftCom |
| Marinescu | Angela | SC SoftCom |
| Antonescu | Elena | SC SoftCom |
| Bischin | Paraschin | SC SoftCom |
| Olaru | Angela | SC SoftCom |
| Ionescu | Gheorghe | SC TimTip |
| Vasilescu | Vasile | SC TimTip |
| Popescu | Ioan | SC TimTip |
| Georgescu | Maria | SC TimTip |
| Marinescu | Angela | SC TimTip |
| Antonescu | Elena | SC TimTip |
| Bischin | Paraschin | SC TimTip |
| Olaru | Angela | SC TimTip |
| Ionescu | Gheorghe | Brasoveanca |
| Vasilescu | Vasile | Brasoveanca |
| Popescu | Ioan | Brasoveanca |
| Georgescu | Maria | Brasoveanca |
| Marinescu | Angela | Brasoveanca |
| Antonescu | Elena | Brasoveanca |
| Bischin | Paraschin | Brasoveanca |
| Olaru | Angela | Brasoveanca |

adică se obţin 8x4 = 32 înregistrări (tabela persoane conţine 8 înregistrări, tabela firme 4 înregistrări)

**Aliasul tabelei este obligatoriu să-l folosim când două tabele conţin coloane cu acelaşi nume. În exemplul anterior coloana prenume nu este obligatoriu să o prefaţăm cu aliasul coloanei, astfel comanda anterioară poate fi scrisă şi astfel:**

**SELECT p.nume, prenume, f.nume**

**FROM persoane p, firme f**

Aşadar, produsul cartezian apare atunci când nu este precizată nici o condiţie privind modul de legare al celor două tabele.

**b) Sintaxa ANSI**Pentru a obţine produsul cartezian, în sintaxa ANSI vom folosi clauza **CROSS JOIN** în cadrul clauzei **FROM** ca în exemplul următor.

**SELECT p.nume, p.prenume, f.nume**

**FROM persoane p CROSS JOIN firme f**

Rezultatul obţinut va coincide cu cel obţinut anterior.

**Exemplul1:**

SELECT name, event\_date, loc\_type, rental\_feeFROM d\_events CROSS JOIN d\_venues;

**II.3.2. Equijoin**

Oare cum procedăm dacă dorim să afişăm pentru fiecare persoană, numele firmei la care lucrează?

**a) Sintaxa Oracle**

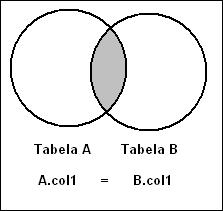
Cum realizăm acest lucru folosind SQL? Simplu. Vom preciza condiţia de egalitate dintre coloanele **IdFirm** din cele două tabele în clauza **WHERE** ca mai jos:

**SELECT p.nume, prenume, f.nume**

**FROM persoane p, firme f**

|  |  |  |
| --- | --- | --- |
| **Nume** | **Prenume** | **Nume** |
| Ionescu | Gheorghe | Brasoveanca |
| Vasilescu | Vasile | SC SoftCom |
| Popescu | Ioan | SC Crisib SA |
| Antonescu | Elena | SC Crisib SA |
| Bischin | Paraschin | SC SoftCom |
| Olaru | Angela | Brasoveanca |

**WHERE p.idfirm = f.idfirm**



**b) Sintaxa ANSI**

În cazul sintaxei ANSI lucrurile se complică uşor. În principal equijoinul se realizează folosind opţiunea **NATURAL JOIN** în cadrul clauzei **from** astfel:

**SELECT nume, prenume, nume**

**FROM persoane NATURAL JOIN firme**

**Însă această comandă nu afişează nici o linie.** De ce? Pentru că **NATURAL JOIN**-ul leagă cele ***două tabele pe toate coloanele cu nume comun din cele două tabele***. Adică, comanda anterioară este echivalentă cu următoarea comandă scrisă folosind sintaxa Oracle:

**SELECT p.nume, prenume, f.nume**

**FROM persoane p, firme f**

**WHERE p.idfirm = f.idfirm AND p.nume=f.nume**

ori nu are nici un sens să punem condiţia ca numele firmei (**f.nume**) să coincidă cu numele persoanei (**p.nume**).

Reguli de folosire a opţiunii **NATURAL JOIN**:

  tabelele sunt legate pe toate coloanele cu nume comun

  coloanele cu nume comun trebuie să aibă acelaşi tip

  în clauza **SELECT** coloanele comune celor două tabele NU vor fi prefaţate de aliasul tabelei.

Pentru a lega două tabele folosind sintaxa ANSI dar condiţia ***de egalitate să fie pusă doar pe anumite coloane*** (nu pe toate coloanele cu nume comun ci doar pe o parte din acestea) se va folosi în loc de **NATURAL JOIN** clauza **JOIN**, iar coloanele pe care se face joinul se precizează în opţiunea **USING**. Astfel comanda pentru afişarea firmelor la care lucrează fiecare angajat se scrie astfel:

**SELECT p.nume, prenume, f.nume**

**FROM personae p JOIN firme f**

**USING (IdFirm)**

**Restricţii la folosirea clauzei JOIN cu clauza USING:**

  în clauza **USING** se trec în paranteză, separate prin virgulă, numele coloanelor pe care se va face joinul

  coloanele din clauza **USING** trebuie să aibă acelaşi tip în cele două tabele

* Permite utilizarea lui **where** pentru a limita randurile.

**Clauza ON**

* Dacă în cele două tabele nu există coloane cu acelaşi nume, sau coloanele cu nume comun au tipuri diferite în cele două tabele, se va folosi clauza **JOIN** în conjuncţie cu **ON**.
* În clauza **ON** pe poate trece orice condiţie de join între cele două tabele(>,<,between)
* Permite utilizarea lui **where** pentru a limita randurile.

**SELECT p.nume, prenume, f.nume**

**FROM persoane p JOIN firme f**

**ON (p.IdFirm=f.IdFirm)**

**Rulati si explicati utilizarea instructiunilor de mai jos:**

**1)**

**Select \* from** d\_play\_list\_items;

Select \* from d\_track\_listings

SELECT event\_id, song\_id, cd\_number

FROM d\_play\_list\_items NATURAL JOIN d\_track\_listings

WHERE event\_id = 105;

2)

SELECT first\_name, last\_name, event\_date, description

FROM d\_clients NATURAL JOIN d\_events;

3)

SELECT client\_number, first\_name, last\_name, event\_date

FROM d\_clients JOIN d\_events

USING (client\_number);

4)

SELECT client\_number, first\_name, last\_name, event\_date

FROM d\_clients JOIN d\_events

USING (client\_number)

WHERE last\_name = ‘Peters’;

5)

SELECT e.last\_name as "EMP", m.last\_name as "MGR"

FROM employees e JOIN employees m

ON (e.manager\_id = m.employee\_id);

6)

SELECT e.last\_name as "EMP", m.last\_name as "MGR"

FROM employees e JOIN employees m

ON (e.manager\_id = m.employee\_id)

WHERE e.last\_name like 'H%';